Crystal and Cavity Characterization

Introduction to digital Low-Level Radio Frequency Controls in Accelerators

Lab 1 and Lab 2

Shreeharshini Dharanesh Murthy, Dmitry Teytelman, Dan Wang, Qiang Du

US Particle Accelerator School
Janurary 23 – 27, 2023

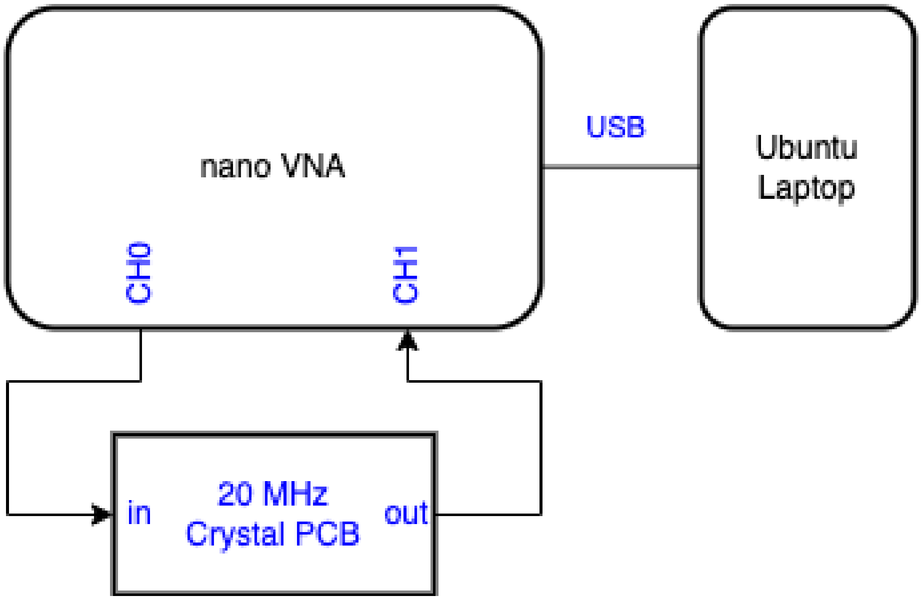
Contents

1 Introduction

In this lab we are going to measure the frequency response using the provided `nanoVNA` network analyzer.

NanoVNA is able to be controlled via USB serial interface from PC. There are sample scripts in this directory.

1.1 Hardware setup



Toggle the power switch of `nanoVNA` to `ON`.

1.2 Software setup

Pre-requirement:

```
$ cd python
$ pip3 install -r requirements.txt
```

Usage of drive `nanovna.py`:

- Plot reflection LOGMAG.

  ```
  $ ./nanovna.py -p -P 0 -S 39.7e6 -E 40.3e6
  ```

- Plot transmission LOGMAG.

  ```
  $ ./nanovna.py -p -P 1 -S 39.7e6 -E 40.3e6
  ```

1.3 Reference

- NanoVNA repository
- NanoVNA manual

2 Exercise

2.1 Crystals:

1. Assuming the Nano-VNA is calibrated (2-port calibration) with center of 20 MHz and span of 1 kHz, captures live data from Nano-VNA.
2. Calculate the 3dB bandwidth, Quality factors (Q_0 and Q_L), and coupling factor.

2.2 Cavities:

1. For a given cavity, find it's resonance frequency.
2. Assuming the Nano-VNA is calibrated (2-port calibration) with center of resonance frequency and required span, capture live data from Nano-VNA.
3. Calculate the 3dB bandwidth, Quality factors (Q_0 and Q_L), and coupling factor.

3 Reference Capture S21 response using `nanoVNA`

Note: the following example uses a 40.0MHz crystal instead the provided 20.00MHz crystal.

Please analysis the data using mathematical concepts taught in the class, and you may compare with results with the `skrf` functions like shown below.

[1]:
```
%matplotlib inline
from nanovna import NanoVNA
from matplotlib import pyplot as plt
import skrf as rf
rf.stylely()
```

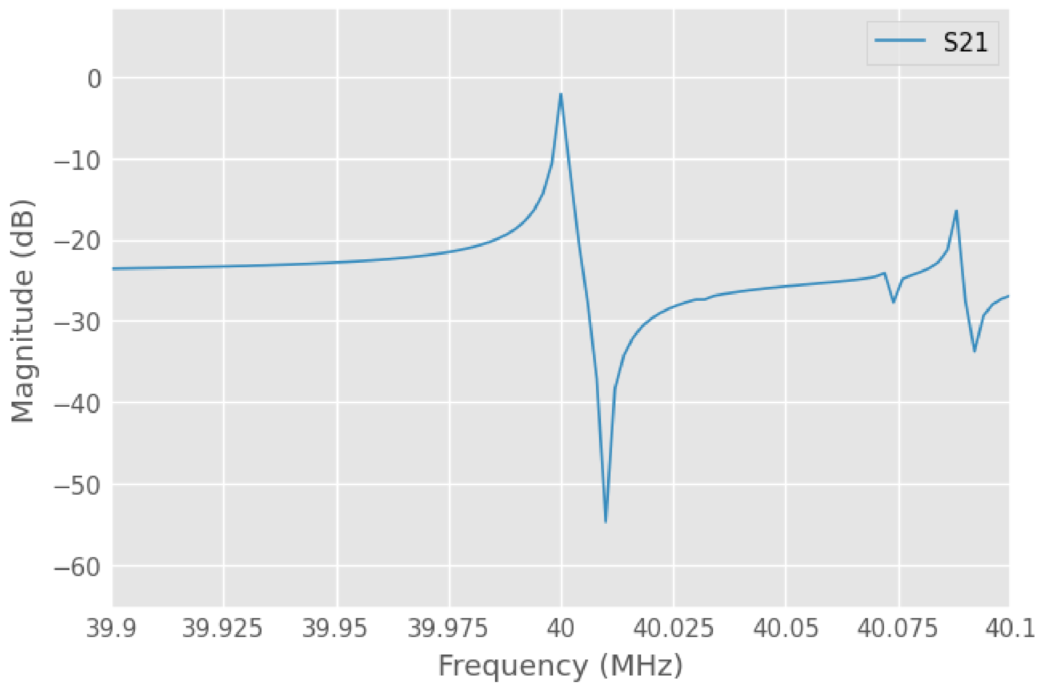
[2]:
```
# define freq range
start_f, stop_f = 39.9e6, 40.1e6

# create instance and connect to the device
nv = NanoVNA()
nv.set_sweep(start_f, stop_f)
nv.fetch_frequencies()
```

[3]:
```
s11 = nv.data(0)
s21 = nv.data(1)
```

[4]:
```
resonator = nv.skrf_network(s21)
```

[5]:
```
resonator.plot_s_db(label='S21')
```



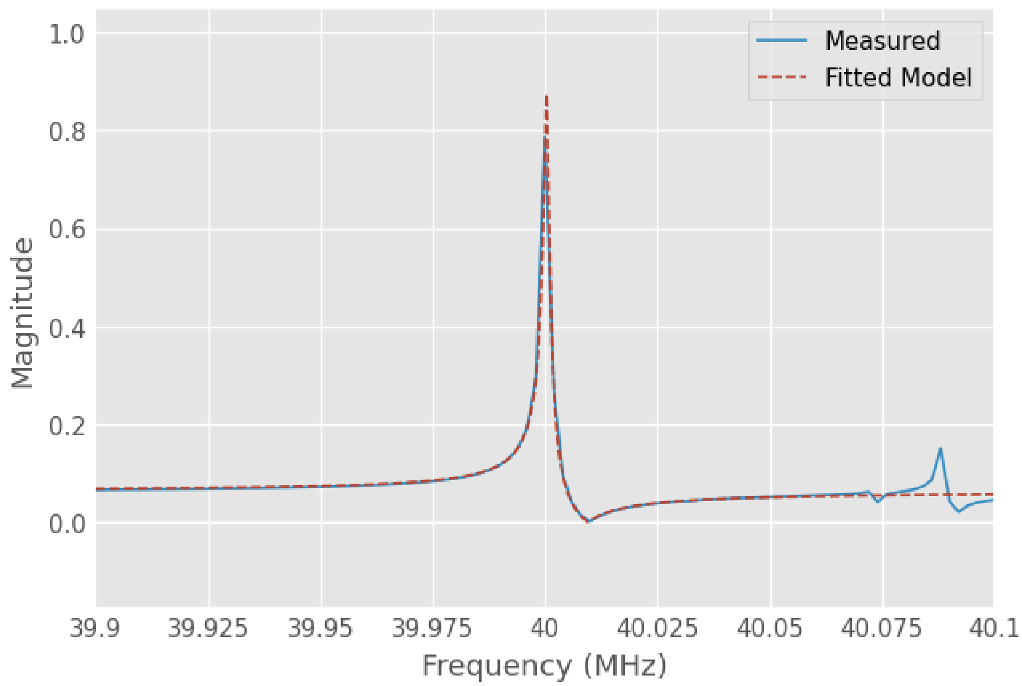
[6]:
```
Q = rf.Qfactor(resonator, res_type='transmission')
res = Q.fit()
```

[8]:
```
Q
```

[8]: Q-factor of Network None. (fitted: f_L=40.000MHz, Q_L=30424.697)

[9]:
```
new_freq = rf.Frequency(start_f / 1e6, stop_f / 1e6, npoints=5001,␣
  ↪unit='MHz')
fitted_network = Q.fitted_network(res, frequency=new_freq)
```

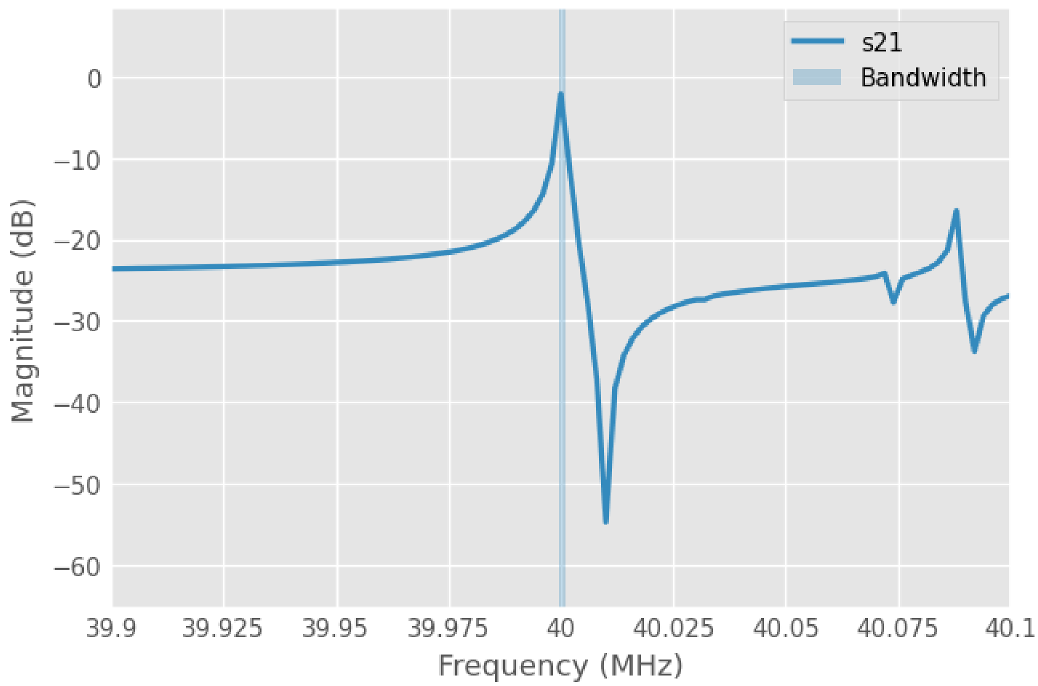
[10]:
```
resonator.plot_s_mag(label='Measured', lw=1)
fitted_network.plot_s_mag(label='Fitted Model', lw=1, ls='--')
```



[11]:
```
print(f'Bandwidth: {Q.BW:.3f} Hz')
```

```
Bandwidth: 1314.734 Hz
```

[13]:
```
fig, ax = plt.subplots()
resonator.plot_s_db(label='s21', lw=2, ax=ax)
ax.axvspan(xmin=Q.f_L-Q.BW/2, xmax=Q.f_L+Q.BW/2, alpha=0.3,␣
 ↪label='Bandwidth')
ax.legend();
# ax.set_xlim(Q.f_L-Q.BW*2, Q.f_L+Q.BW*2)
```


s21
Bandwidth
0
−10
−20
−30
−40
−50
−60
Magnitude (dB)
39.9
39.925
39.95
39.975
40
40.025
40.05
40.075
40.1
Frequency (MHz)